TE&ET'06

I Congreso de Tecnología en Educación y Educación en Tecnología.
8 al 11 de Agosto de 2006, La Plata, Argentina

Desarrollo e implementación de un Intérprete de Pseudocódigo para la Enseñanza de Algorítmica Computacional

Autores

Loyarte Horacio. FICH-Universidad Nacional del Litoral, hloyarte@fich.unl.edu.ar
Novara , Pablo. FICH-Universidad Nacional del Litoral, zaskar_84@yahoo.com.ar

Resumen

A la problemática del ingreso irrestricto en las universidades de nuestro país, debemos sumar la pobre calidad de la enseñanza en el nivel medio/polimodal, lo cual produce una combinación claramente negativa para un adecuado proceso de enseñanza-aprendizaje en los primeros años de las carreras de grado. Este problema se traduce en porcentajes muy altos de deserción en el primer año de estudios de nuestras UUNN. Este es el caso de la a carrera de Ingeniería Informática en la UNL de Santa Fe, que además agrega la problemática de un ingreso masivo de estudiantes en primer año. En esta carrera, durante el primer cuatrimestre del primer año se desarrolla la asignatura Fundamentos de Programación, la cual introduce a los estudiantes a los conceptos de programación en base a un laboratorio de lenguaje ANSI/ISO C++. Con la problemática mencionada, la enseñanza de la lógica requerida para la resolución de problemas mediante programas, sumada a la sintaxis algo críptica (para principiantes) de C++ dificulta el aprendizaje y el cumplimiento de los objetivos de la materia. Por lo expuesto, la cátedra de la asignatura introdujo desde el año 2002 una serie de innovaciones destinadas a revertir algunos de estos inconvenientes y mejorar los resultados del proceso educativo. Entre los cambios se destaca el uso de un pseudocódigo en español en las 4 primeras unidades de la asignatura que permitiera desarrollar en los estudiantes los conceptos de lógica de programación y el uso de ciertas estructuras básicas de control y de datos en forma independiente de un lenguaje de alto nivel. Este trabajo describe las características del desarrollo de un intérprete para dicho pseudocódigo y su empleo en la enseñanza de algorítmica computacional.

Palabras claves: Intérprete – Pseudocódigo – Algorítmica - Computacional

Descripción de la problemática

Situación general

Son muchas los motivos que confluyen a los ya conocidos resultados negativos del proceso educativo que culminan los adolescentes de nuestro país al completar su formación del nivel medio/polimodal. No es el objeto de este trabajo, pero podemos mencionar como causas de este problema aspectos culturales y psicológicos de los jóvenes: su relación con los adultos, la ausencia de límites en el hogar y en la escuela, la ausencia de modelos que pregonen la cultura del esfuerzo, etc. Pero la realidad nos dice a diario que los ingresantes a los primeros años de las carreras universitarias, tienen serios problemas de adaptación, lo cual afecta a la calidad del sistema educativo universitario.

Estos inconvenientes se traducen en altos porcentajes de deserción en el primer año de la mayoría de las carreras, en pobres desempeños académicos, en problemas para el cuerpo docente a la hora de

decidir y desarrollar contenidos, requerimientos de infraestructura universitaria para un número de alumnos que diminuye en forma drástica a los pocos meses de iniciar el cursado, etc.

El caso de estudio
En la Facultad de Ingeniería y Ciencias Hídricas de la Universidad Nacional del Litoral, se desarrolla la carrera de Ingeniería Informática desde el año 1999. A la problemática general descripta en el epígrafe anterior, se suma un ingreso masivo de alumnos que dificulta el normal desarrollo de contenidos y la retención de alumnos. La asignatura Fundamentos de Programación del primer cuatrimestre plantea como contenidos un laboratorio de lenguaje ANIS/ISO C++ para desarrollar los conceptos básicos de algorítmica computacional y programación. Esta materia se complementa con la del 2do cuatrimestre: Programación Orientada a Objetos.
La experiencia recogida de varios años de desarrollar la asignatura, los magros resultados de las primeras evaluaciones y la heterogeneidad de cada una de las sucesivas cohortes que ingresaban a la carrera, llevó a pensar en alternativas superadoras del proceso de aprendizaje. Más aún, considerando que en la asignatura se tratan contenidos claves dentro de la disciplina que da sustento a la carrera.

Propuesta Didáctica y Metodológica

La enseñanza de algorítmica computacional
El uso de lenguaje ANSI/ISO C++ para una materia inicial de la programación, presenta algunas dificultades para estudiantes inexpertos que deben aprender varios conceptos relativos al diseño de algoritmos y paralelamente lidiar con cuestiones de implementación de las soluciones propuestas relativas a un lenguaje de programación: sintaxis, compilación, mensajes de errores en inglés, depuración, etc. Por otro lado, el lenguaje C++ es la herramienta de desarrollo profesional de mayor difusión en el ámbito profesional y admite operaciones y sintaxis basadas en criterios de eficiencia, pero carece de transparencia para un estudiante que se inicia en las ciencias de la computación.
Esas dificultades y ralentización del aprendizaje de las primeras unidades de la asignatura, fue claramente advertido por el personal de la cátedra, por lo cual se resolvió investigar y proponer alternativas superadoras que facilitaran el aprendizaje de los alumnos.

Propuestas de solución
Entre las propuestas para mejorar el aprendizaje significativo, se propuso: a) utilizar una plataforma virtual de aprendizaje, que permitiera realizar actividades complementarias a las de las clases tradicionales y registrar las acciones de los estudiantes, ya que debido a la masividad, la identificación de cada individuo era imposible de realizar; b) proponer para la enseñanza de algorítmica computacional un pseudocódigo en español, con reglas sintácticas sencillas y básicas, que permitiera concentrar al alumno en la lógica para la resolución de problemas mediante el diseño y la construcción de algoritmos y facilitara el aprendizaje y uso posterior de un lenguaje de alto nivel. El presente trabajo nos ocupa del desarrollo del ítem (b).

Desarrollo de un intérprete de pseudocódigo en español

Premisas para el desarrollo del software
Para facilitar el aprendizaje en el diseño de algoritmos para resolver problemas, se propuso adoptar un pseudocódigo en español que respondiera ala siguientes premisas:

- Sintaxis sencilla

- Manejo de las estructuras básicas de control
- Solo 3 tipos de datos básicos: numérico, caracter /cadenas de caracteres y lógico (verdadero-falso).
- Estructuras de datos. arreglos
- Plataforma: que pueda ejecutarse indistintamente en Linux y Windows.
- Diseño del software orientado a objetos.

El desarrollo del intérprete

Las características del pseudolenguaje fueron propuestas en 2001 por el responsable de la asignatura Fundamentos de Programación (Horacio Loyarte) de la carrera de Ingeniería Informática de la FICH-UNL y co-autor de este trabajo. El desarrollo de la primer versión del intérprete fue realizada en 2003 por Pablo Novara, coautor de este documento y alumno de la carrera de Ingeniería Informática, con el objeto de ser presentado como proyecto final de la materia Programación Orientada a Objetos. Fue desarrollado utilizando el paradigma de la Programación Orientada a Objetos, empleando inicialmente Borland C++ Builder 4.0 por ser una herramienta de tipo RAD que facilitaba el diseño de interfaces y poseer un excelente sistema de ayuda en línea. El softaware consta de dos componentes principales:

1. El intérprete propiamente dicho, que trabaja en modo consola y se limita a analizar y ejecutar un algoritmo tomado desde un archivo o informar debidamente al usuario los errores de lógica y/o sintaxis que se encuentren en el mismo.
2. La Interfaz Gráfica de Usuario (GUI), que consiste en un editor integrado, documentación y ayuda en línea, algunos ejemplos, y ciertas facilidades para la ejecución y corrección de los algoritmos.

Aunque se pensó desde un principio que se liberaría bajo licencia GPL, fue a mediados de 2004 cuando el proyecto se publico en SourceForge.Net y se comenzaron a hacer las adaptaciones necesarias: fue portado a GNU/Linux utilizando el toolkit para el desarrollo de interfaces Gimp Toolkit (GTK), y se adaptó el código existente al ANSI/ISO C++ para que pudiera ser compilado en cualquier plataforma con el compilador libre GCC. Esto, junto con su inclusión en la recopilación "Actividades Educativas con Software Libre" de la colección educ.ar, amplió considerablemente el número de usuarios y generó una provechosa retroalimentación por parte de los mismos acerca del funcionamiento del intérprete, lo que posibilitó su depuración y perfeccionamiento. Actualmente el proyecto continúa su desarrollo y se pueden encontrar regularmente nuevas versiones con las correcciones sugeridas por dichos usuarios en http://pseint.sourceforge.net.

Algunas características de la sintaxis

i. Estructura general de un algoritmo en PseInt

Todo algoritmo en PseInt tiene la estructura general que se indica a la derecha.

En la figura 1 se observa el sencillo entorno de desarrollo que presenta la herramienta de desarrollo para comenzar a escribir algoritmos en pseudocódigo en español.

```
Proceso SinTitulo
    accion 1;
    accion 1;
      .
      .
      .
    accion n;
FinProceso
```

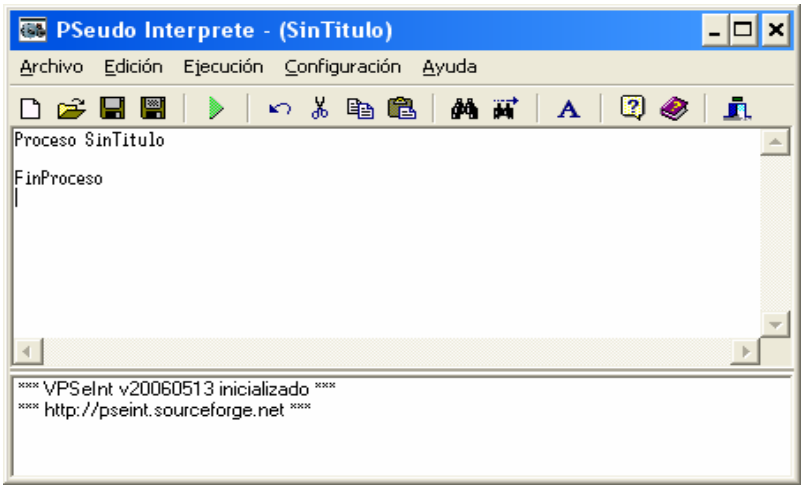


Figura 1. Interfaz del entorno de desarrollo de PseInt.

ii. Tipos de datos

PseInt solo admite tres tipos básicos de datos: numérico, caracter o cadena de caracteres y lógico. Para el tipo numérico no se distinguen flotantes de enteros. El tipo caracter incluye los caracteres del código ASCII así como los datos de tipo string. El tipo lógico admite como únicas constantes: verdadero , falso.

iii. Operadores

La tabla 1 exhibe la totalidad de los operadores de este lenguaje reducido.

Símbolo	Significado	Ejemplo
Operadores Relacionales		
>	Mayor que	x>12
<	Menor que	'Ana' < 'Carlos'
=	Igual que	R = 234
<=	Menor	Nombre<=' Juan'
>=	Mayor	z>= x+1
<>	Distinto	y <> 100
Operadores Lógicos		
&	Conjunción	(a<2) & (r=100)
|	Disyunción	(ape='Lopez') | (c=0)
~	Negación	~ (x>200)
Operadores Algebraicos		
+	Suma	a+1
-	Resta	x-y
*	Producto	2*a
/	Cociente	x/y
^	potencia	m^3
Operador de Asignación		
<-	Asignación	x <- a+1;
Operador de Subíndice		
[ ]	Subindicación de arreglos	x[i]<- 34;

Tabla 1. Operadores de PseInt

iv. Estructuras de control

PseInt dispone de las estructuras de control clásicas, presentes en cualqiuer lenghuaje de programación: If-then, While, Repeat-Unitil (do-while), Case (switch), For; todas ellas expresadas en español. Los recuadros siguientes incluyen la sintaxis de cada estuctura de control del pseudolenguaje.

```
Si  <exp. lógica>
      entonces A
      [sino B]
FinSi
```

```
Mientras <exp. lógica> Hacer
      acción A;
      acción B;
      acción C;
FinMientras
```

```
Según V  Hacer
      1:   A:
      2:   B;
      3,4:  C;
            .
            .
      n:   Q;
     [De otro modo: R;]
FinSegún
```

```
Repetir
      acción P ;
      acción Q ;
      acción R ;
 Hasta que <exp. lógica>
```

v. Estructuras de datos

Por cuestiones operativas relacionadas con el desarrollo de la asignatura, PseInt solo amite la estructura de datos arreglo, de uno, dos o más dimensiones.

Obsérvese en la figura 2 la codificación de un algoritmo que opera con una arreglo de hasta 100 elementos numéricos, y su correspondiente ejecución en la pantalla en modo consola.

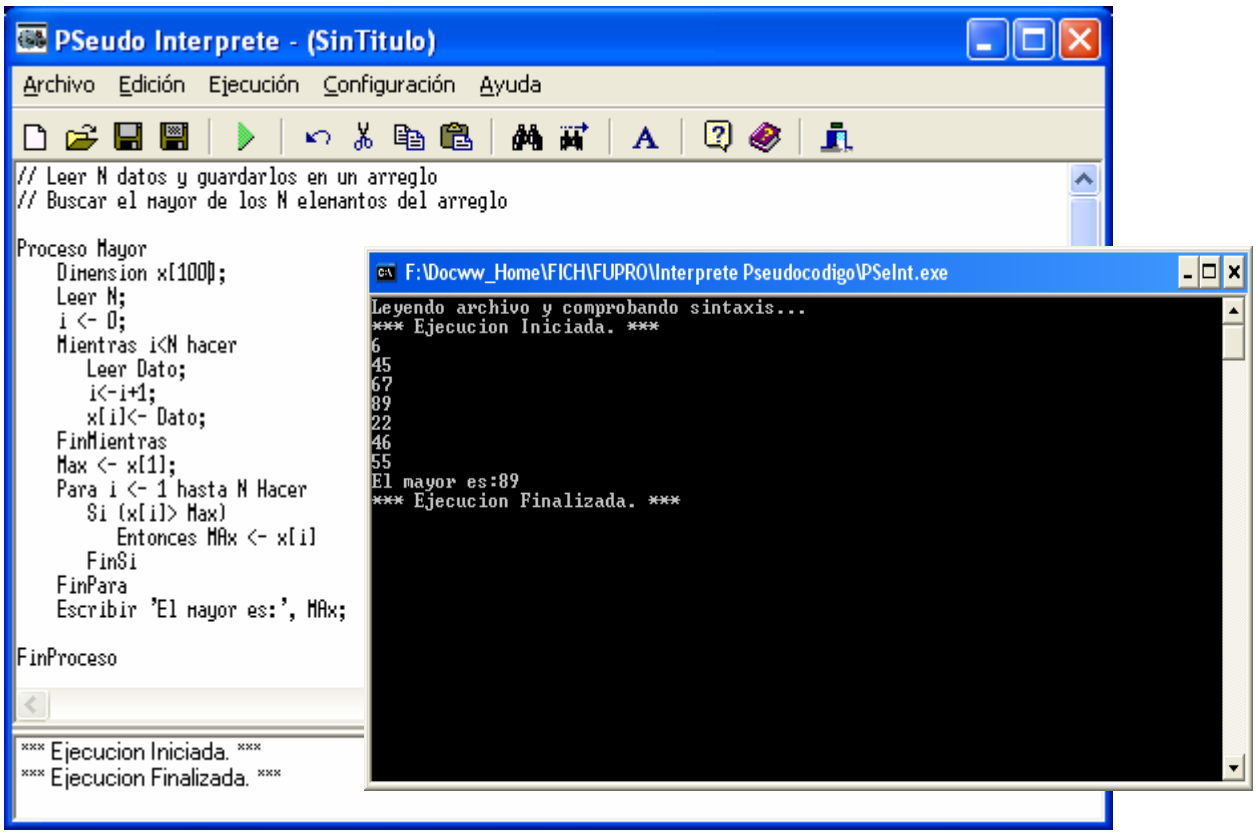


Figura 2. Código y ejecución de un algoritmo usando arreglos

Implementación dentro del diseño curricular

La inclusión del diseño y codificación de algoritmos empleando pseudocódigo en la asignatura Fundamentos de Programación (FICH-UNL), fue propuesta a partir de 2003. Esta inclusión abarca 4 primeras unidades de la asignatura. Luego de estos 4 módulos los realizan una evaluación parcial. El objetivo de esa primer parte de la materia es desarrollar en los alumnos los conceptos de la lógica de programación para la resolución de problemas, aplicando estructuras básicas de control y estructuras de datos sencillas, empleando en la codificación un lenguaje algorítmico procedimental dentro del paradigma de la programación estructurada. Dicho lenguaje es el pseudocódigo (PseInt) propuesto en este trabajo. Luego de esa primer parte de la asignatura, las unidades 5,6,..,10 sn empleadas en el desarrollo de un laboratorio de programación ANSI/ISO C++. La ventaja del empleo del pseudolenguaje en las primeras unidades se aprecia al tratar con la sintaxis e implementación de programas en C++, pues los estudiantes manejan conceptos de algorítmica computacional e integran la nueva sintaxis con naturalidad, afianzando sus conocimientos de programación.

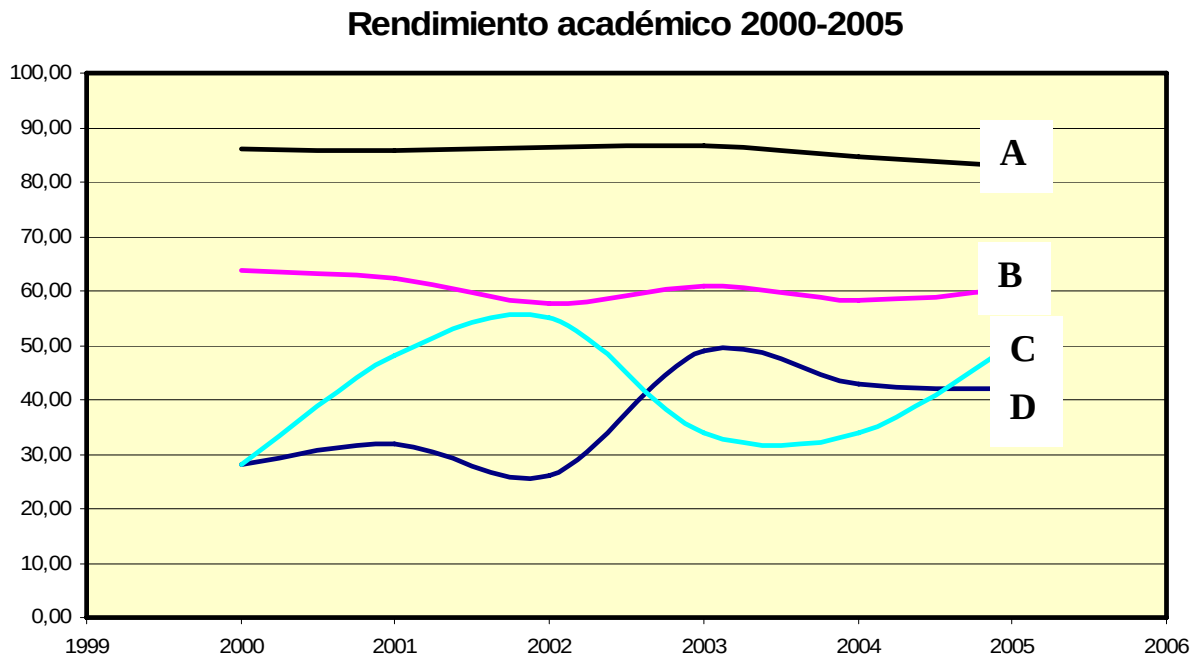
Resultados

A continuación se vuelcan en la tabla 2 algunos parámetros resultantes del desempeño académico de los estudiantes de Ingeniería Informática en la asignatura empleada como testigo (Fundamentos de Programación).

Año	Porcentaje de regulares sobre total de alumnos **(D)**	Promedio notas regulares **(B)**	Promedio notas promovidos **(A)**	Porcentaje de Promovidos sobre regulares **(C)**
2000	28,00	63,68	86,17	28,00
2001	32,00	62,33	85,83	48,00
2002	26,00	57,77	86,40	55,00
2003	49,00	60,85	86,71	34,00
2004	43,00	58,38	84,62	34,00
2005	42,00	60,40	83,01	50,00

Tabla 2. Resultados de los últimos 6 años

Obsérvese que a partir de 2003 se implementa el uso de pseudocódigo y los parámetros de calidad del aprendizaje medidos desde el rendimiento de los estudiantes no se han modificado (curvas A, B y C), pero sin embargo la curva D presenta un significativo incremento de partir de la implementación de la propuesta didáctica incrementándose el % de regulares en más de 10 puntos porcentuales.



Conclusión

En poblaciones con similar número de individuos (250 alumnos por año), hemos podido detectar a lo largo del análisis de 6 años de desarrollo de una asignatura inicial de programación en una carrera de Informática/Sistemas, que el % de alumnos regulares se incrementa si se comienza por el aprendizaje de la lógica de programación a través de un pseudolenguaje (pseudocódigo) de sintaxis sencilla. También se ha detectado que el resto de la población que cursaba la asignatura normalmente no ha experimentado una mejora significativa. La presencia de mayor cantidad de regulares también evita el incremento de la deserción en el primer año de estudios de la carrera. Sería de sumo interés realizar un seguimiento de las poblaciones estudiadas a lo largo de los años superiores de la carrera, para verificar si esta mejora/retención es efímera en el tiempo o permite rescatar futuros profesionales.

Bibliografía

Braunstein-Gioia. “Introducción a la Programación”. Eudeba.. 1987.

Cátedra Fundamentos de Programación. Apuntes de cátedra. Facultad de Ciencia y Tecnología. Universidad Autónoma de Entre Ríos. 2003.

Loyarte, Horacio. Apuntes de cátedra: “Fundamentos de Programación”. FICH-UNL. 2004, 2005, 2006.

Deitel y Deitel. “C++ Cómo programar”. 4ta Ed. Pearson Educación- Prentice Hall. 2003.

Stroustrup B. “El Lenguaje de Programación C++”. 3ra Ed. Addison Wesley. 1998.

Joyanes Aguilar Luis. “Fundamentos de Programación”. MCGRAW-HILL. 2003.

Cairó Osvaldo. “Metodología de la programación”. AlfaOmega Editores. 2003.

Pérez T. J., Gojenola K. “Una Experiencia para la Mejora en los Resultados de las Prácticas de Programación", III Congreso Iberoamericano de Educación Superior en Computación, Concepción, Chile, 1994.

Pinninghoff, Contreras, Polyméris y otros. “Una Proposición para la Enseñanza de Lenguajes de Programación en Ingeniería”. III Congreso Iberoamericano de Educación Superior en Computación. Chile. 1995.

Baeza Yates, Ricardo. “Diseñemos todo de nuevo: Reflexiones sobre la computación y su enseñanza”. En: Revista Colombiana de Computación. Vol. 1. No. 1. 2000. ISSN 1657-2831.

Blom, Martin; Nordby, Eivind; Brunstrom, Anna. “Teaching Semantic Aspects of OOP. Fifth Workshop on Pedagogies and Tools for Assimilating Object Oriented Concepts OOPSLA'01”. 2001. http://www.cs.umu.se/ jubo/Meetings/OOPSLA01/Program.html

Brosgol, Benjamin M. A “Comparison of Ada and Java as a Foundation Teaching Language”. Ada Core Technologies. 2000.

Felleisen, M., Findler, R. B., Flatt M., Krishnamurthi, S. “How to Design Programs: An Introduction to Computing and Programming”. The MIT Press. 2001. http://www.htdp.org/2002-09-22/Book/

Felleisen, M., Findler, R. B., Flatt M., Krishnamurthi, S. “The Structure and Interpretation of the Computer Science Curriculum”. Functional and Declarative Programming in Education (FDPE02). 2002. http://people.cs.uchicago.edu/ robby/publications/papers/htdp-sicp-fdpe2002.pdf

Robinson, Peter. “From ML to C via Modula-3 an approach to teaching programming”. 1994. http://www.cl.cam.ac.uk/˜pr/mlm3/mlm3